TOAD i ShyShark
Dokumentacja wstępna

Wstęp

TOAD (Toad Owns All Data) jest rozproszoną siecią 'peer-to-peer'. Jej topologia odpowiada głównym założeniem tzw. *scale-free network* (http://en.wikipedia.org/wiki/Scale-free_network). Pozwala na udostępnianie, wyszukiwanie i pobieranie dowolnych zasobów (pliki, strumienie audio/video, etc.), których obsługę implementują klienci sieci.

Podstawową funkcją TOAD jest udostępnianie mechanizmów wyszukiwania i rozgłaszania zasobów. Za ich pobieranie/wysyłanie odpowiadają klienci, wykorzystując odpowiednie rozszerzenie protokołu. Z tego powodu TOAD można nazwać meta-protokołem.

Implementacja w założeniu ma być zgodna przede wszystkim z platformami UNIX'owymi. Jednak dzięki bibliotekom *cygwin*, będzie utrzymywany także *port* dla systemów rodziny MS Windows.

Elementy sieci

Założeniem jest stworzenie połączenia informacyjnego między wieloma równoprawnymi węzłami sieci (Leaf) przy pomocy kilku pośrednich węzłów nadrzędnych, skupiających na sobie wiele połączeń (Hub). Natomiast połączenia służące do wymianych danych binarnych są bezpośrednie.

Node (węzeł) – Leaf lub Hub

Leaf (liść) – zwyczajny klient sieci

- łączy się z przynajmniej 1 hubem
- wszelkie zapytania o zasoby przekazuje do Huba

Hub (węzeł centralny) – klient, do którego łączą się inne węzły w celu obsługi zapytań.

- łączy się z przynajmniej 1 hubem
- utrzymuje połączenia z innymi hubami tak, by zachować spójność sieci (by dowolne dwa węzły mogły w każdej chwili się skomunikować)
- pozawala na dołączonej pewnej liczby klientów (>= 1)
- funkcjonalność Huba jest nadzbiorem funkcjonalności Leaf'a

Komunikacja pomiędzy węzłami

Komunikacja opiera się na protokole TCP. Strona nawiązująca połączenie wysyła dane w standardowej postaci (nagłówek binarny, właściwy nagłówek, dane binarne).

Właściwy nagłówek ma postać skompresowanego XML.

Schemat strumienia danych

wersja	długość nagłówka właściwego	właściwy nagłówek XML	dane binarne
2 bajty	2 bajty	< 2^16 bajtów	bez ograniczeń

Istnieją dwa podstawowe rodzaje wymienianych informacji:

a) Żądania

Strona przyjmująca wiadomość musi na nią odpowiedzieć w odpowiedni sposób i zgodnie z posiadanymi danymi

b) Informacje

Strona przyjmująca potwierdza odebranie danych wysyłając prostą wiadomość (zawsze identyczną).

W przypadku, kiedy któryś z nagłówków jest błędny, strona przyjmująca połączenie musi wysłać odpowiednią informację o błędzie (w normalnym formacie).

Struktura sieci i przepływ wiadomości

a) Połączenia leaf-to-leaf

Tworzone tylko w celu wymiany konkretnych (znanych obu stronom) zasobów.

b) Połączenia leaf-to-hub

Wyszukiwanie zasobów odbywa się przez huby - leaf wysyła do hubów, u których się zarejestrował, zapytanie, które huby dalej przekazują między sobą. Hub dowolny po odebraniu zapytania odpytuje przyłączone leafy (połączenie typu żądanie) i jeśli odnalazł wyszukiwany zasób wysyła tę informację do klienta, który rozpoczął wyszukiwanie (połączenie typu informacja). Wtedy klient szukający może bezpośrednio połączyć się z węzłem udostępniającym zasób.

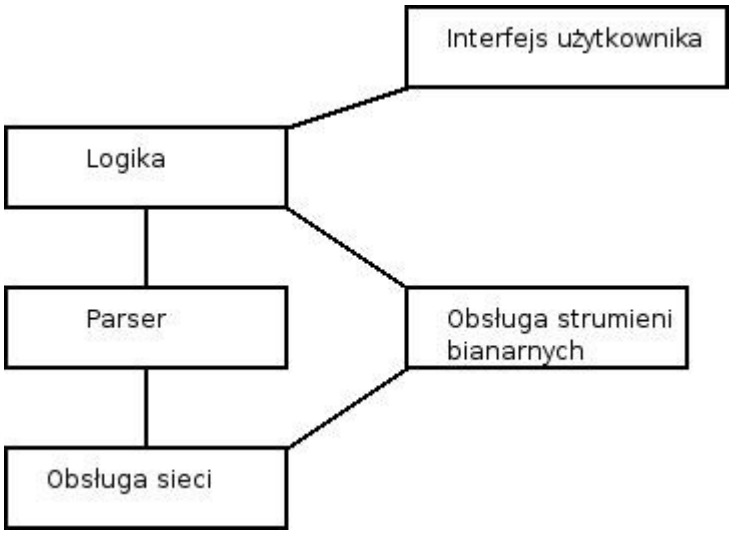
c) Połączenia hub-to-hub

Hub łącząc się między sobą tworzą szkielet sieci, umożliwiający wyszukiwanie zasobów. Każdy hub w normalnych warunkach powinien łączyć się z co najmniej 2 innymi, aby zachować spójność sieci.

Aplikacja kliencka – ShyShark

Implementuje rozszerzenie protokołu TOAD umożliwiające wymianę plików między dowolnymi węzłami.

Architektura aplikacji pozwala na łatwą rozbudowę funkcjonalności sieciowej i implementację własnego interfesu użytkownika.



Obsługa sieci

Warstwa odpowiadająca za komunikację sieciową z innymi węzłami. Tłumaczy strumień TCP z postaci binarnej na nagłówek XML i strumień danych binarnych.

Parser

- Parsuje nagłówek XML, weryfikując jego poprawność i tłumacząc na postać prostych zdarzeń zrozumiałych przez warstwę logiki.
- Tłumaczy proste wywołanie z warstwy logicznej na pełen nagłówek XML.

Obsługa strumieni binarnych

Pośredniczy w wymianie danych między warstwami.

Logika

Kontroluje wszystkie połączenia, podejmuje decyzje o wysyłaniu wiadomości i odpowiedzi do innych węzłów. Zarządza wymianą zasobów (pliki) na danym kliencie.

Interfejs użytkownika

Wysyła do warstwy logiki żądania użytkownika (np: wyszukiwanie, pobieranie) i odbiera informacje o zdarzeniach w aplikacji (np: nawiązano połączenie, otrzymano wyniki wyszukiwania).

Założenia dotyczące interfejsu użytkownika

Interfejs nie zależy od detali implementacyjnych całej aplikacji. Pozwala to na uniezależnienie procesu produkcyjnego interfejsu oraz łatwiejszą jego rozbudowę.

Interfejs użytkownika wymienia informacje z warstwą logiczną aplikacji. Taka konstrukcja umożliwia pełną kontrolę nad wydawanymi poleceniami i zachowaniem aplikacji.

Założeniem jest opracowanie protkołu komunikacji interfejsu użytkownika z “jądrem” aplikacji. Takie podejście umożliwia nie tylko utrzymanie modularności projektu, ale nawet stworzenie dwóch osobnych aplikacji: interfejsu i aplikacji właściwej.Druga korzyść, to duża niezależność od platformy.

Pierwsza wersja interfejsu będzie opierała się o bibliotekę *ncurses* i będzie jeszcze wkompilowana w aplikację. Będzie się składać linii poleceń oraz okna komunikatów.